plex flight profile to be simulated, as well as ambient conditions and deterioration level of the engine. C-MAPSS40k has three actuators: fuel flow, variable stator vanes, and variable bleed valve. The three actuators enable off-nominal operation, which is not possible with simulations that have fuel flow as the sole actuator, since in those simulations the other actuators are implicit and assumed to operate nominally. The simulation is modular to allow users to redesign or replace components such as the engine controller or turbomachinery components without having to modify the rest of the simulation. It also enables the user to view and save any signal in the engine or controller. The package has the capability to create and validate a linear model of the engine at any operating point. Linear models can be used for control design, and C-MAPSS40k lends itself well to implementation and evaluation of advanced control designs as well as to diagnostic and prognostic system development. The simulation can be run in real time and can therefore be integrated into a flight simulator with a pilot in the loop for testing.

C-MAPSS40k fills the need for an easy-to-use, realistic, transient simulation of a medium-size commercial turbofan engine with a representative controller. It is a detailed component level model (CLM) written in the industry-standard graphical MATLAB/Simulink environment to allow for easy modification and portability. At the time of this reporting, no other such model exists in the public domain.

This work was done by Ten-Huei Guo, Thomas Lavelle, and Jonathan Litt of Glenn Research Center and Jeffrey Csank of N&R Engineering and Ryan May of ASRC. Further information is contained in a TSP (see page 1).

Inquiries concerning rights for the commercial use of this invention should be addressed to NASA Glenn Research Center, Innovative Partnerships Office, Attn: Steven Fedor, Mail Stop 4–8, 21000 Brookpark Road, Cleveland, Ohio 44135. Refer to LEW-18624-1.

The Planning Execution Monitoring Architecture

Lyndon B. Johnson Space Center, Houston, Texas

The Planning Execution Monitoring (PEM) architecture is a design concept for developing autonomous cockpit command and control software. The PEM architecture is designed to reduce the operations costs in the space transportation system through the use of automation while improving safety and operability of the system. Specifically, the PEM autonomous framework enables automatic performance of many vehicle operations that would typically be performed by a human. Also, this framework supports varying levels of autonomous control, ranging from fully automatic to fully manual control.

The PEM autonomous framework interfaces with the "core" flight software to perform flight procedures. It can either assist human operators in performing procedures or autonomously execute routine cockpit procedures based on the operational context. Most importantly, the PEM autonomous framework promotes and simplifies the capture, verification, and validation of the flight operations knowledge. Through a hierarchical decomposition of the domain knowledge, the vehicle command and control capabilities are divided into manageable functional "chunks" that can be captured and verified separately. These functional units, each of which has the responsibility to manage part of the vehicle command and control, are modular, re-usable, and extensible. Also, the functional units are self-contained and have the ability to plan and execute the necessary steps for accomplishing a task based upon the current mission state and available resources.

The PEM architecture has potential for application outside the realm of spaceflight, including management of complex industrial processes, nuclear control, and control of complex vehicles such as submarines or unmanned air vehicles.

This work was done by Lui Wang, Bebe Ly, and Alan Crocker of Johnson Space Center; Debra Schreckenghost of Metrica Inc; Stephen Mueller and Bob Phillips of Titan-LinCom Corp.; and David Wadsworth and Charles Sorensen of Lockeed Martin Corp. For further information, contact the Johnson Commercial Technology Office at (281) 483-3809. MSC-23628-1

Jitter Controller Software

Lyndon B. Johnson Space Center, Houston, Texas

Sinusoidal jitter is produced by simply modulating a clock frequency sinusoidally with a given frequency and amplitude. But this can be expressed as phase jitter, frequency jitter, or cycle-to-cycle jitter, rms or peak, absolute units, or normalized to the base clock frequency. Jitter using other waveforms requires calculating and downloading these waveforms to an arbitrary waveform generator, and helping the user manage relationships among phase jitter crest factor, frequency jitter crest factor, and cycle-to-cycle jitter (CCJ) crest factor.

Software was developed for managing these relationships, automatically configuring the generator, and saving test results documentation. Tighter management of clock jitter and jitter sensitivity is required by new codes that further extend the already high performance of space communication links, completely correcting symbol error rates higher than 10 percent, and therefore typically requiring demodulation and symbol synchronization hardware to operating at signal-to-noise ratios of less than one. To accomplish this, greater demands are also made on transmitter performance, and measurement techniques are needed to confirm performance. It was discovered early that sinusoidal jitter can be stepped on a grid such that one can connect points by constant phase jitter, constant frequency jitter, or constant cycle-cycle jitter. The tool automates adherence to a grid while also allowing adjustments off-grid. Also, the jitter can be set by the user on any dimension and the others are calculated. The calculations are all recorded, allowing the data to be rap-

idly plotted or re-plotted against different interpretations just by changing pointers to columns.

A key advantage is taking data on a carefully controlled grid, which allowed a single data set to be post-analyzed many different ways. Another innovation was building a software tool to provide very tight coupling between the generator and the recorded data product, and the operator's worksheet. Together, these allowed the operator to sweep the jitter stimulus quickly along any of three dimensions and focus on the response of the system under test (response was jitter transfer ratio, or performance degradation to the symbol or codeword error rate). Additionally, managing multi-tone and noise waveforms automated a tedious manual process, and provided almost instantaneous decision-making control over test flow. The code was written in LabVIEW, and calls Agilent instrument drivers to write to the generator hardware.

This work was done by Chatwin Lansdowne and Adam Schlesinger of Johnson Space Center. Further information is contained in a TSP (see page 1). MSC- 24814-1

µShell Minimalist Shell for Xilinx Microprocessors

NASA's Jet Propulsion Laboratory, Pasadena, California

µShell is a lightweight shell environment for engineers and software developers working with embedded microprocessors in Xilinx FPGAs. (µShell has also been successfully ported to run on ARM Cortex-M1 microprocessors in Actel ProASIC3 FPGAs, but without project-integration support.) µShell decreases the time spent performing initial tests of field-programmable gate array (FPGA) designs, simplifies running customizable one-time-only experiments, and provides a familiar-feeling command-line interface. The program comes with a collection of useful functions and enables the designer to add an unlimited number of custom commands, which are callable from the command-line. The commands are parameterizable (using the C-based command-line parameter idiom), so the designer can use one function to exercise hardware with different values. Also, since many hardware peripherals instantiated in FPGAs have reasonably simple register-mapped I/O interfaces, the engineer can edit and view hardware parameter settings at any time without stopping the processor.

µShell comes with a set of support scripts that interface seamlessly with Xilinx's EDK tool. Adding an instance of µShell to a project is as simple as marking a check box in a library configuration dialog box and specifying a software project directory. The support scripts then examine the hardware design, build design-specific functions, conditionally include processor-specific functions, and complete the compilation process. For code-size constrained designs, most of the stock functionality can be excluded from the compiled library.

When all of the configurable options are removed from the binary, µShell has an unoptimized memory footprint of about 4.8 kB and a size-optimized footprint of about 2.3 kB. Since µShell allows unfettered access to all processor-accessible memory locations, it is possible to perform live patching on a running system. This can be useful, for instance, if a bug is discovered in a routine but the system cannot be rebooted: µShell allows a skilled operator to directly edit the binary executable in memory. With some forethought, µShell code can be located in a different memory location from custom code, permitting the custom functionality to be overwritten at any time without stopping the controlling shell.

This work was done by Thomas A. Werne of Caltech for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

This software is available for commercial licensing. Please contact Daniel Broderick of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-47495.

Software Displays Data on Active Regions of the Sun

Lyndon B. Johnson Space Center, Houston, Texas

The Solar Active Region Display System is a computer program that generates, in near real time, a graphical display of parameters indicative of the spatial and temporal variations of activity on the Sun. These parameters include histories and distributions of solar flares, active region growth, coronal mass ejections, size, and magnetic configuration.

By presenting solar-activity data in graphical form, this program accelerates, facilitates, and partly automates what had previously been a time-consuming mental process of interpretation of solar-activity data presented in tabular and textual formats. Intended for original use in predicting space weather in order to minimize the exposure of astronauts to ionizing radiation, the program might also be useful on Earth for predicting solar-wind-induced ionospheric effects, electric currents, and potentials that could affect radio-communication systems, navigation systems, pipelines, and long electric-power lines.

Raw data for the display are obtained automatically from the Space Environment Center (SEC) of the National Oceanic and Atmospheric Administration (NOAA). Other data must be obtained from the NOAA SEC by verbal communication and entered manually. The Solar Active Region Display System automatically accounts for the latitude dependence of the rate of rotation of the Sun, by use of a mathematical model that is corrected with NOAA SEC active-region position data once every 24 hours. The display includes the date, time, and an image of the Sun in Hα light overlaid with latitude and longitude coordinate lines, dots that mark locations of active regions identified by NOAA, identifying numbers assigned by NOAA to such regions, and solar-region visual summary (SRVS) indicators associated with some of the active regions.

Each SRVS indicator is a small pie chart containing five equal sectors, each